Import a GPG Private Key and Encrypt/Decrypt Messages Using GPG

807 Jisna Joseph February 3, 2026 Security Compliances 950

GPG Private Key Import and Message Decryption Guide

Overview: This guide provides step-by-step instructions for importing GPG keys, decrypting encrypted messages, and encrypting messages. GPG stands for **GNU Privacy Guard** which is used to protect information using encryption.

1. Install GPG

Step 1(A): Update the package repository and install GPG

```
root@server:# sudo apt update

root@server:# sudo apt install gnupg -y
```

Step 1(B): Verify that GPG is installed correctly by checking the version:

```
root@server:# gpg --version
```

2. Generate new private GPG key

Step 2(A): A private GPG key is a secure file used to decrypt encrypted messages. Generate a key if you don't already have a private key. This key will be used to decrypt messages. If a private key already exists, proceed to **Step 3**.

Run the below command (for GPG ≥ 2.1.17):

```
root@server:# gpg --full-generate-key
```

For older GPG version run below command:

```
root@server:# gpg --default-new-key-algo rsa4096 --gen-key
```

Step 2(B): Follow the prompts:

Select key type: Specify the kind of key you want, or press Enter to accept the default.(RSA and RSA)

Key size: Specify the key size you want, or press Enter to accept the default.

Key expiration: Type your preferred duration (e.g., 1y for 1 year) or 0 for no expiration

Verify that your selections are correct.

User ID information: Enter your name and a verified email address

Comment: Optional

Passphrase: Type a strong passphrase to protect the private key

Step 2(C): After generation, confirm the key:

```
root@server:# gpg --list-secret-keys --keyid-format=long
```

You should see an entry similar to:

```
sec rsa4096/XXXXXXXXXXXXXXX
uid Your Name <your@email.com>
```

3. Import an Existing Private GPG Key

Step 3(A): If an existing private key file is already available (for example: support_private_key.asc), upload it to the server and import it using the following command:

```
root@server:# gpg --import support_private_key.asc
```

The private key file name may vary. Replace **support_private_key.asc** with the actual file name provided.

Step 3(B): When prompted, enter the private key passphrase to complete the import process

Expected Output:

Confirmation that the secret key is imported
Associated user ID (email/name)

Step 3(C): List the secret keys to confirm that the private key was imported successfully:

```
root@server:# gpg --list-secret-keys --keyid-format LONG
```

You should see an entry similar to:

```
sec rsa4096/XXXXXXXXXXXXXXX
uid Ezeelogin Support <support@ezeelogin.com>
```

4. Decrypt an Encrypted Message File

Step 4(A): If an encrypted file is received (for example, message.asc), use the following command to decrypt it:

```
root@server:# gpg --decrypt message.asc
```

Step 4(B): When prompted, enter the private key passphrase to proceed with decryption.

Step 4(C): After successful decryption, the decrypted content will be displayed directly in the terminal.

Step 4(D): Run below command to save the decrypted content into a file.

```
root@server:# gpg --decrypt message.asc > decrypted_message.txt
```

5. Decrypt an Encrypted Message from Email (Text)

Step 5(A): Copy the entire encrypted block **from** `-----BEGIN PGP MESSAGE-----` **to** `-----END PGP MESSAGE-----`)

Step 5(B): Create a file and paste the encrypted message into it:

```
root@server:# nano encrypted_message.asc
```

Step 5(C): Save and close the file.

Step 5(D): Decrypt the message using the following command:

```
root@server:# gpg --decrypt encrypted_message.asc
```

6. Import a Public GPG Key

Step 6(A): A public GPG key is used to encrypt messages so that only the intended recipient can decrypt them using their private key. After receiving the public key file (for example: public_key.asc), upload it to the server and import it using the following command:

```
root@server:# gpg --import public_key.asc
```

7. Encrypt and Send a Message

Step 7(A): To encrypt a message for a recipient using their public key, run:

```
root@server:# echo "Sensitive message here" | gpg --encrypt --armor -r recipient@email.com > message.asc
Send the content of message.asc via email (Outlook, Gmail, etc.).
```

Step 7(B): Send the contents of message.asc via email (Outlook, Gmail, etc.).